

Dynamic Programming: Example 2

Longest Common Subsequence

Problem: Let $x_1x_2\dots x_m$ and $y_1y_2\dots y_n$ be two sequences over some alphabet. (We assume they are strings of characters.) Find a longest common subsequence (LCS) of $x_1x_2\dots x_m$ and $y_1y_2\dots y_n$.

Example: $x_1x_2x_3x_4x_5x_6x_7x_8 = \mathbf{b a c b f f c b}$
 $y_1y_2y_3y_4y_5y_6y_7y_8y_9 = \mathbf{d a b e a b f b c}$
 $z_1z_2z_3z_4z_5 = \mathbf{b a b f c}$ is an LCS (shown below).

Subproblems: Find an LCS of $x_1x_2\dots x_i$ and $y_1y_2\dots y_j$ ($0 \leq i \leq m, 0 \leq j \leq n$).

Optimal substructure: If $z = z_1z_2\dots z_p$ is a LCS of $x_1x_2\dots x_m$ and $y_1y_2\dots y_n$, then at least one of these most hold.

- i) $x_m = y_n$, and $z_1z_2\dots z_{p-1}$ is an LCS of $x_1x_2\dots x_{m-1}$ and $y_1y_2\dots y_{n-1}$,
- ii) $x_m \neq y_n$, and $z_1z_2\dots z_p$ is an LCS of $x_1x_2\dots x_{m-1}$ and $y_1y_2\dots y_n$,
- iii) $x_m \neq y_n$, and $z_1z_2\dots z_p$ is an LCS of $x_1x_2\dots x_m$ and $y_1y_2\dots y_{n-1}$.

Let c_{ij} = length of LCS of $x_1x_2\dots x_i$ and $y = y_1y_2\dots y_j$.

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ 1 + c[i-1,j-1] & \text{if } x_i = y_j, \\ \max(c[i-1,j], c[i,j-1]) & \text{if } x_i \neq y_j. \end{cases}$$

$$b[i,j] = \begin{cases} "\uparrow_" & \text{if } x_i = y_j, \\ "\uparrow" & \text{if } x_i \neq y_j \text{ and } c[i-1,j] \geq c[i,j-1], \\ "\leftarrow" & \text{if } x_i \neq y_j \text{ and } c[i-1,j] < c[i,j-1]. \end{cases}$$

We compute the $c[i,j]$ and $b[i,j]$ in order of increasing $i+j$, or alternatively in order of increasing i , and for a fixed i , in order of increasing j .

Example: $x_1x_2x_3x_4x_5x_6x_7x_8 = \mathbf{b a c b f f c b}$
 $y_1y_2y_3y_4y_5y_6y_7y_8y_9 = \mathbf{d a b e a b f b c}$

Row i , column j of the table below contains the value of $c[i,j]$ followed (except when $i = 0$ or $j = 0$) by that of $b[i,j]$

	0	1	2	3	4	5	6	7	8	9
		d	a	b	e	a	b	f	b	c
0	0	0	0	0	0	0	0	0	0	0
1 b	0	0 ↑	0 ↑	1 ↖	1 ←	1 ↑	1 ↖	1 ←	1 ↖	1 ←
2 a	0	0 ↑	1 ↖	1 ↑	1 ↑	2 ↖	2 ←	2 ←	2 ←	2 ←
3 c	0	0 ↑	1 ↑	1 ↑	1 ↑	2 ↑	2 ↑	2 ↑	2 ↑	3 ↖
4 b	0	0 ↑	1 ↑	2 ↖	2 ←	2 ↑	3 ↖	3 ←	3 ↖	3 ↑
5 f	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↖	4 ←	4 ←
6 f	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↖	4 ↑	4 ↑
7 c	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↑	4 ↑	5 ↖
8 b	0	0 ↑	1 ↑	2 ↖	2 ↑	2 ↑	3 ↖	4 ↑	5 ↖	5 ↑

We can compute each table element in constant time, so the entire table takes $\Theta(mn)$ time.

We can write down an LCS by starting in the lower right corner and following the arrows backward.

Whenever we reach a square containing a " \nwarrow ", say in row i and column j , we insert the character $x_i = y_j$ at the beginning of the subsequence.

	0	1	2	3	4	5	6	7	8	9
		d	a	b	e	a	b	f	b	c
0	0	0	0	0	0	0	0	0	0	0
1 b	0	0 ↑	0 ↑	1 ↖	1 ←	1 ↑	1 ↖	1 ←	1 ↖	1 ←
2 a	0	0 ↑	1 ↖	1 ↑	1 ↑	2 ↖	2 ←	2 ←	2 ←	2 ←
3 c	0	0 ↑	1 ↑	1 ↑	1 ↑	2 ↑	2 ↑	2 ↑	2 ↑	3 ↖
4 b	0	0 ↑	1 ↑	2 ↖	2 ←	2 ↑	3 ↖	3 ←	3 ↖	3 ↑
5 f	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↖	4 ←	4 ←
6 f	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↖	4 ↑	4 ↑
7 c	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↑	4 ↑	5 ↖
8 b	0	0 ↑	1 ↑	2 ↖	2 ↑	2 ↑	3 ↖	4 ↑	5 ↖	5 ↑

An LCS is $x_1x_2x_4x_5x_7 = y_3y_5y_6y_7y_9 = \mathbf{babfc}$.

This computation takes $\Theta(m+n)$ time.